[Your Name] - [Your Student ID]

MIS710 Machine Learning in Business

Estimating Game Ratings for Play Quest Conquer

Table of Contents

[Executive Summary 1](#_Toc174264735)

[Business Understanding 2](#_Toc174264736)

[Data Understanding and Preparation 2](#_Toc174264737)

[Data Understanding 2](#_Toc174264738)

[Data Exploration 3](#_Toc174264739)

[Data Visualization 5](#_Toc174264740)

[Data Preparation 9](#_Toc174264741)

[Insights Gained 11](#_Toc174264742)

[Machine Learning Approach 12](#_Toc174264743)

[Model and Performance Metrics 12](#_Toc174264744)

[Discussion of Model Pros and Cons 13](#_Toc174264745)

[References 14](#_Toc174264746)

[Appendices 14](#_Toc174264747)

# Executive Summary

Machine learning analysis for Play Quest Conquer (PQC) focused on enhancing game development, acquisition, and promotion strategies by examining game configurations, user engagement, and popularity indicators. Key findings revealed a correlation between average playtime and ratings, the impact of game complexity on user satisfaction, and the significance of popularity metrics like the number of owners and traders. A supervised model was developed to predict game ratings, with performance assessed through relevant metrics. The report offers actionable insights for PQC to improve game quality, align offerings with user preferences, and boost engagement.

# Business Understanding

The project aimed to predict the average ratings of games on the Play Quest Conquer (PQC) platform, providing actionable insights for market research and development teams. By analyzing a dataset encompassing game configurations, user engagement, and popularity indicators, the project sought to identify key factors influencing user ratings. These insights would guide PQC in refining their game development, acquisition, and promotion strategies to boost user satisfaction, engagement, and overall game ratings.

PQC's primary business objectives included enhancing user satisfaction, optimizing game offerings, increasing engagement and retention, and strategizing effective game promotion. The project focused on answering critical business questions about the factors that most influence game ratings, the impact of game attributes on user satisfaction, and the predictability of game ratings based on these features. The success of the analysis was measured by the accuracy of the predictive model and the actionable insights generated to inform PQC's strategic decisions.

# Data Understanding and Preparation

## Data Understanding

The dataset is loaded from a CSV file, containing various features such as Released\_Year, Min\_Players, Average\_Complexity, and Average\_Rating, among others. These features relate to board games, aiming to predict the Average\_Rating of the games. Let's break down the dataset features you mentioned in more detail:

**Game\_ID**: A unique identifier assigned to each game.

**Game\_Name**: The title or name of the game.

**Released\_Year**: The year when the game was officially released.

**Game\_Type**: Classification of the game, such as BaseGame or PremiumGame.

**Age\_Category**: The recommended age range for players.

**Min\_Players**: The minimum number of players required to play the game.

**Max\_Players**: The maximum number of players allowed in the game.

**Average\_Complexity**: The average rating of the game's complexity.

**Complexity\_Raters**: The number of individuals who rated the game's complexity.

**Average\_Play\_Time**: The average duration, in minutes, players spend on the game.

**Owner\_Number**: The count of individuals who own the game.

**Trader\_Number**: The number of people who have traded or exchanged the game.

**HighInterest\_Number**: The number of people who have shown a strong interest in playing the game.

**Interest\_Number**: The number of people who have shown a general interest in playing the game.

**Average\_Rating**: The average rating given to the game by players.

**Rater\_Number**: The number of people who have rated the game.

**Comment\_Number**: The total number of comments received about the game.

## Data Exploration

The first step involves exploring the dataset by displaying the first and last five rows, along with the total number of tuples (rows) and features (columns). The data types of each feature are inspected, with special attention to numerical columns. Data exploration is a crucial step in understanding the dataset before diving into deeper analysis or modeling. Here's how I can approach this:

1.Load the Dataset

Load the dataset into a DataFrame using a tool like Python's pandas library.

# #Import the dataset

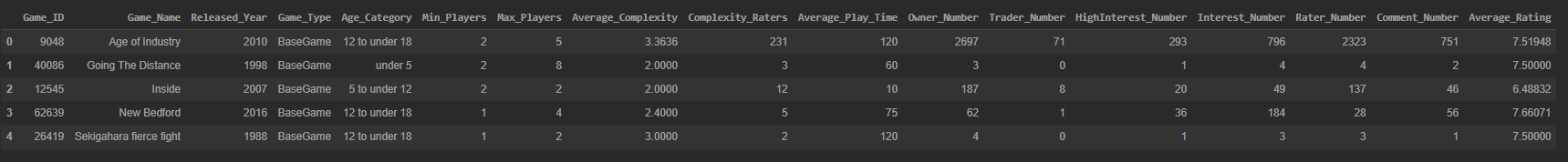
dataframe = pd.read\_csv('/content/drive/My Drive/Colab Notebooks/PQC/PQC\_data.csv')

2. Display the First and Last Five Rows

This gives an initial look at the data, helping I understand its structure and identify any immediate issues such as missing values.

#View the first five rows from the data frame

dataframe.head()



#View the last five rows from the data frame

dataframe.tail()

3.Count the Total Number of Rows and Columns

This gives you a sense of the dataset's size, which is important for understanding the scale of analysis required.

#Get the data set tuple and features

print('No of tuples: ', dataframe.shape[0],'| No of features: ',dataframe.shape[1])

4. Inspect the Data Types of Each Feature

Understanding the data types helps I know how to handle each feature. For instance, numerical data can be used for statistical analysis, while categorical data may need encoding.

#check attribute data type

print(dataframe.dtypes)

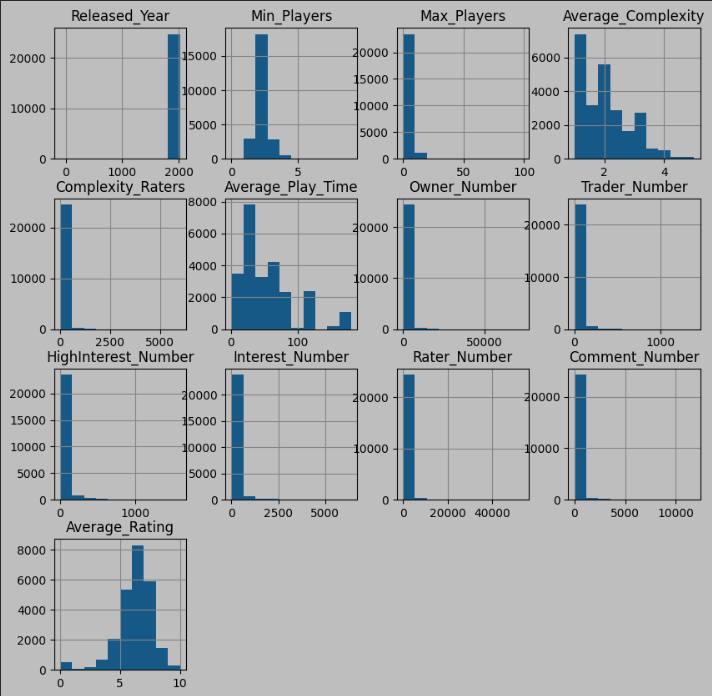
5. Focus on Numerical Columns

Numerical columns are often the focus of predictive modeling. I inspect them for their statistical properties.

# Visualizing the numerical features of the dataset using histograms to analyze the distribution of those features in the dataset

rcParams['figure.figsize'] = 10, 10

dataframe[['Released\_Year', 'Min\_Players', 'Max\_Players', 'Average\_Complexity', 'Complexity\_Raters', 'Average\_Play\_Time', "Owner\_Number", "Trader\_Number", "HighInterest\_Number", "Interest\_Number", "Rater\_Number", "Comment\_Number", "Average\_Rating"]].hist()

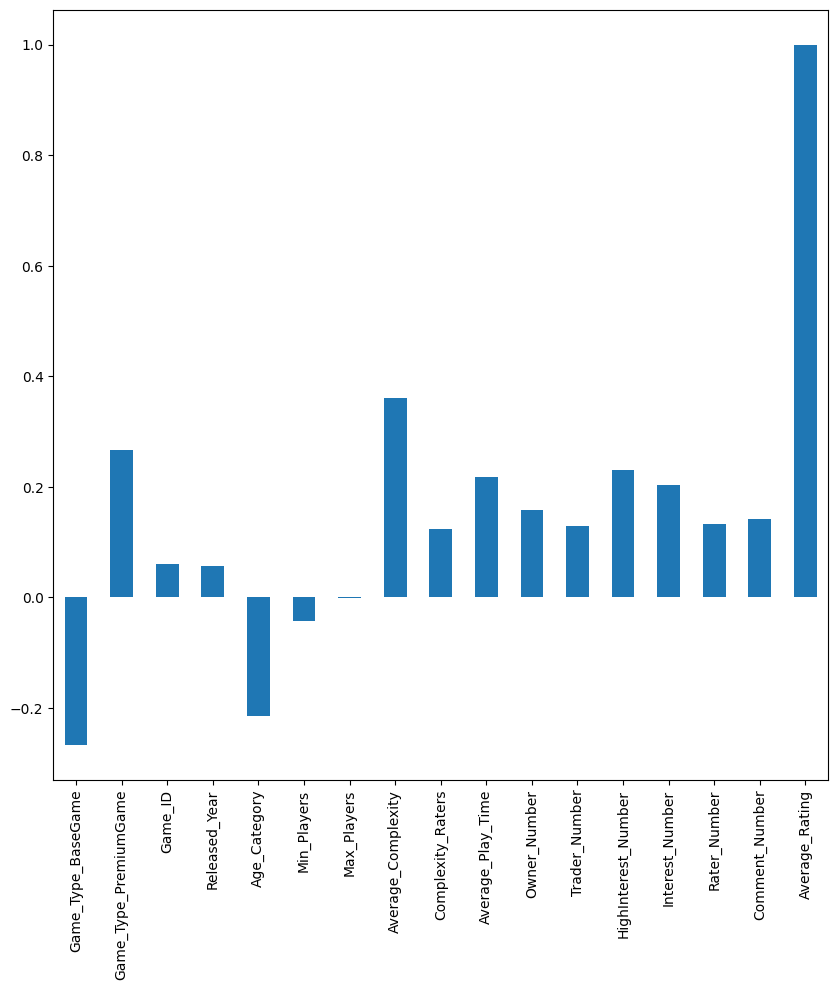


## Data Visualization

Data visualization is a powerful tool for understanding the underlying patterns and relationships in a dataset. Histograms are plotted for numerical features to analyze their distributions. A correlation matrix is visualized using a heatmap to identify relationships between features and their correlation with the target variable (Average\_Rating). Boxen plots are used to explore the distribution of values and detect any outliers in the dataset. Here's how I can approach visualizing the dataset:

1.Plot Histograms for Numerical Features

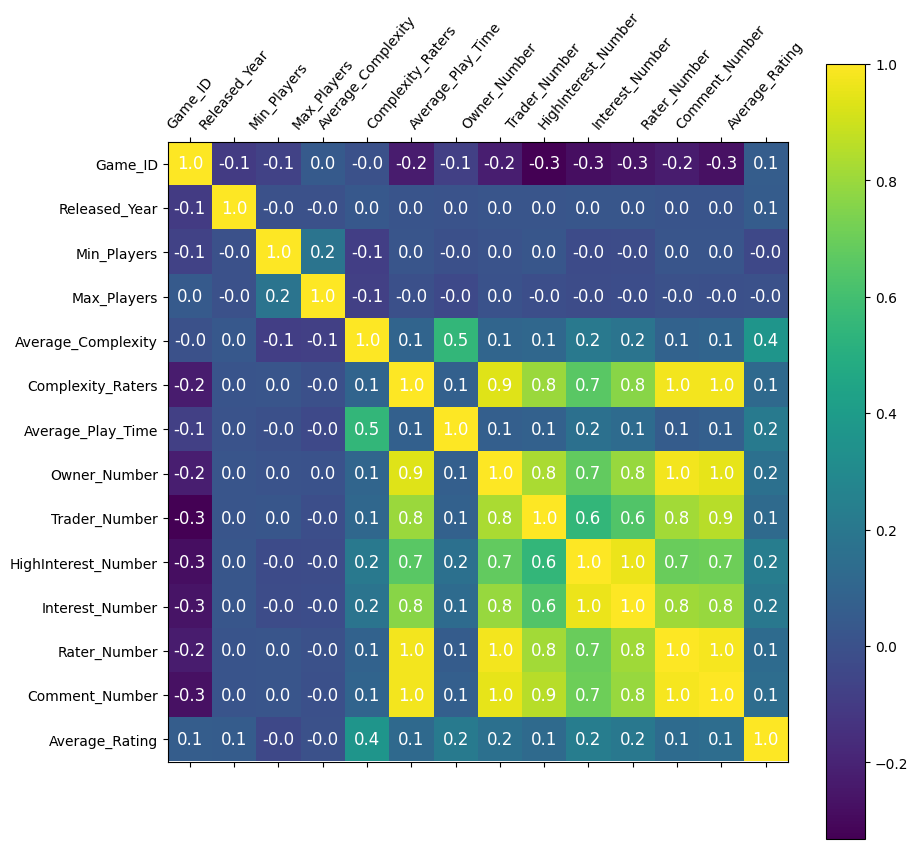
Histograms are useful for understanding the distribution of numerical features. They allow you to see the frequency of different values within a feature and can help identify any skewness or unusual patterns.



Check if the data is normally distributed, skewed, or if there are any anomalies. For example, features like Average\_Complexity might show how complex games are typically rated, and Released\_Year could show trends over time.

2. Visualize the Correlation Matrix Using a Heatmap

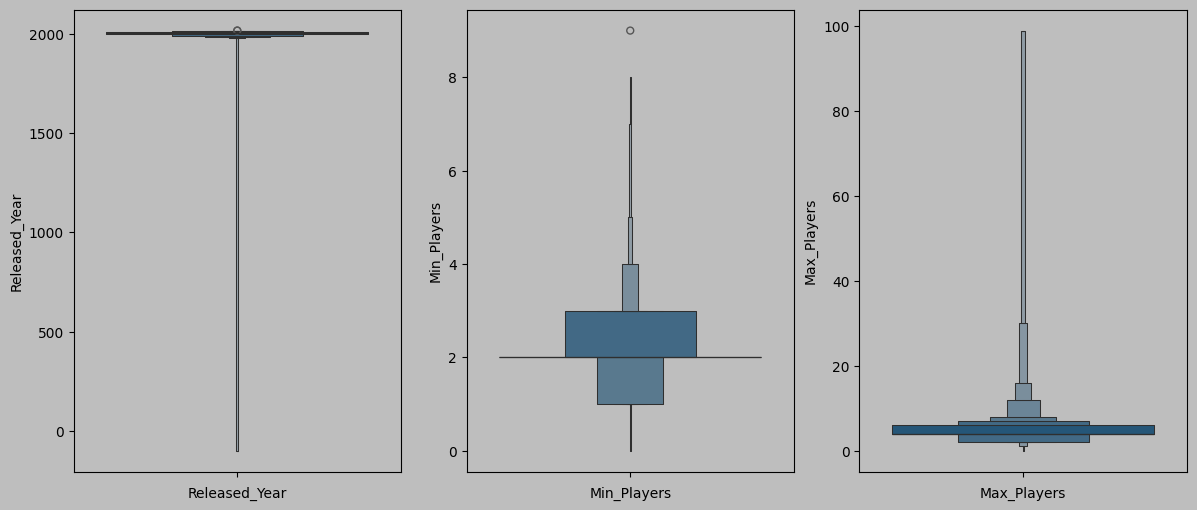
A correlation matrix helps to understand the relationship between numerical features. A heatmap can be used to visualize this matrix, where the color intensity represents the strength of the correlation.

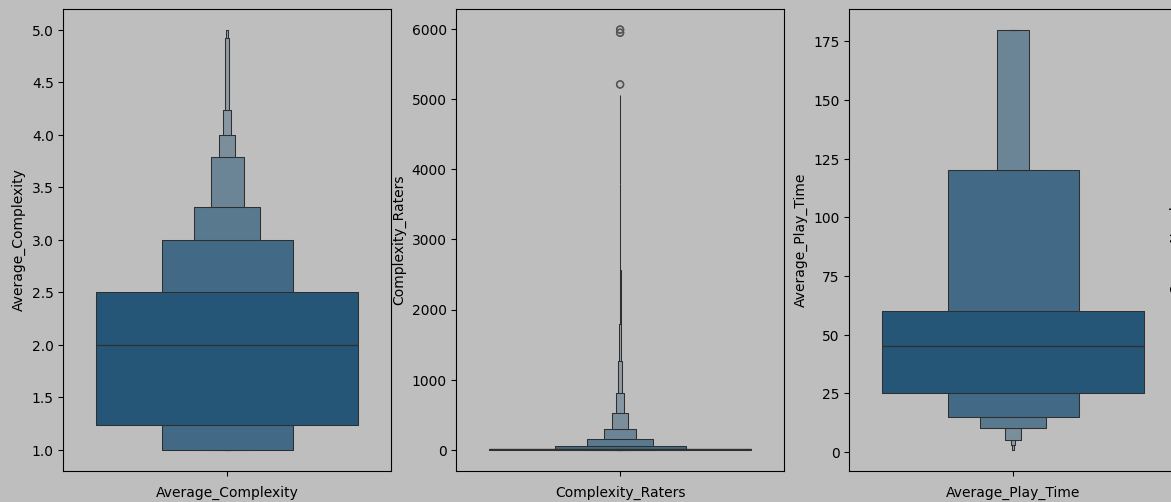


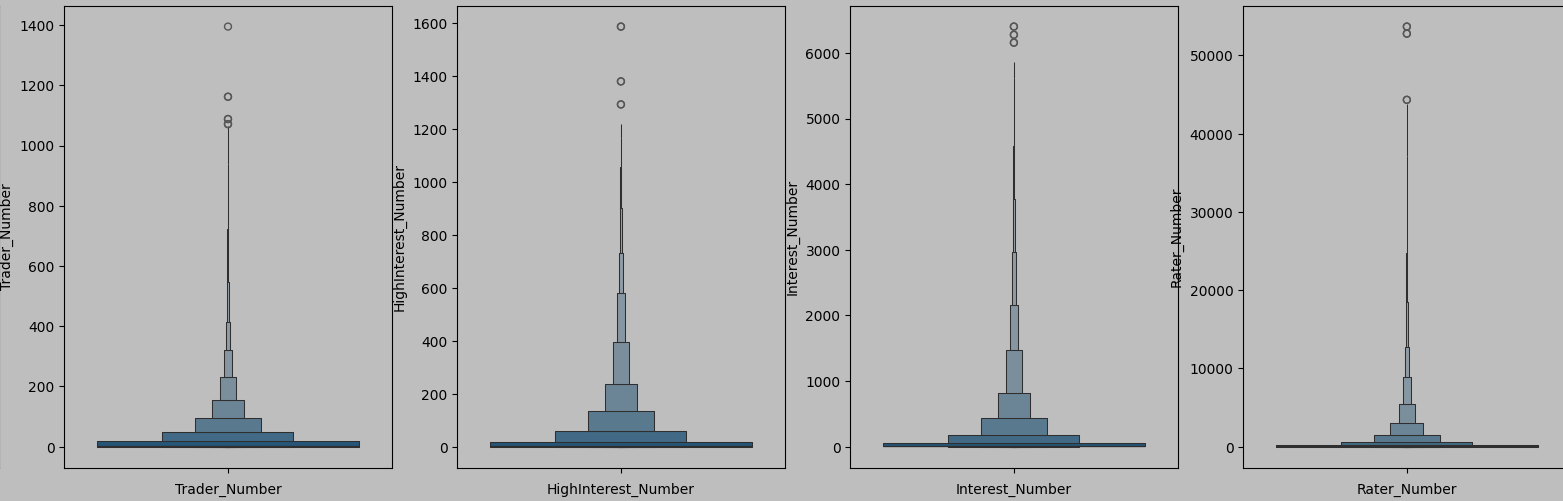
Focus on how features like Average\_Complexity or Min\_Players correlate with the target variable Average\_Rating. Strong positive or negative correlations indicate that these features could be important predictors.

3.Use Boxen Plots to Explore Distribution and Detect Outliers

Boxen plots are an advanced type of box plot that are particularly useful for large datasets, as they can show more details about the distribution, especially in the tails. They help in identifying outliers and understanding the distribution of features.







Identify any outliers that may affect your analysis. For example, if Average\_Rating has outliers, you might want to investigate further to understand why those ratings are so different.

## Data Preparation

This phase involves cleaning and transforming the data to make it suitable for modeling. Here's how I can approach it:

1.Dropping Unnecessary Columns

Some columns may not contribute to the predictive power of your model or might introduce noise. In this case, Game\_Name and Max\_Players are dropped.

#Game\_Name and Max\_Players column is unneccesary.so it can be dropped

dataframe.drop(["Game\_Name","Max\_Players"],axis=1,inplace=True)

The variable Game\_Name typically serves as a unique identifier for each game and does not contribute to the prediction of Average\_Rating. As such, it lacks predictive value and may be excluded from the model. Similarly, Max\_Players, while providing information on the capacity of a game, may also be deemed irrelevant to predicting Average\_Rating. Therefore, if it does not show a significant correlation with the target variable, it might be dropped to streamline the model and improve its accuracy.

2. Handling Missing Values

Before proceeding with encoding or other transformations, it's essential to assess and handle missing values in the dataset.

#Found what is the presentage of data is missing from the dataframe

total\_missing\_value = dataframe.isnull().sum().sort\_values(ascending=False)

present\_1 = dataframe.isnull().sum()/dataframe.isnull().count()\*100

present\_2 = (round(present\_1, 1)).sort\_values(ascending=False)

missing\_value = pd.concat([total\_missing\_value, present\_2], axis=1, keys=['Total Missing Value', '%'])

missing\_value.head(15)

Columns with a high percentage of missing values can significantly impact the analysis and model performance. If a column has an excessive amount of missing data, it may be more practical to drop it from the dataset rather than trying to impute or address the missing values. This is because columns with too many missing values can introduce noise, reduce the reliability of imputation methods, and complicate the analysis. For this step, we focus on identifying and removing these columns to ensure the dataset remains clean and manageable for further analysis.

3. Encoding Categorical Data

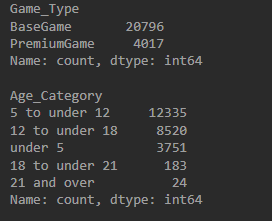
Firstly I try to view object details across this code.

#View count of cells with object data type

for d in dataframe.select\_dtypes(include = 'object'):

    print(dataframe[d].value\_counts())

    print("")



a. Ordinal Encoding for Age\_Category

Age\_Category might represent ordered categories (e.g., "12 to under 18", "under 5"). Ordinal Encoding converts these into numerical values reflecting their order.

# Encode 'Age\_Category' column with numerical values using label encoding

ordinal\_encoder = OrdinalEncoder()

dataframe[['Age\_Category']] = ordinal\_encoder.fit\_transform(dataframe[['Age\_Category']])

dataframe[['Age\_Category']].head(10)

This encoding is suitable for categorical variables with a clear order, ensuring that the numerical representation reflects the inherent order of categories.

b. One-Hot Encoding for Game\_Type

Game\_Type likely has multiple categories (e.g., "PremiumGame", "BaseGame"). One-Hot Encoding creates binary columns for each category, ensuring that the model doesn't assume any ordinal relationship between these categories.

# Encode 'Game\_Type' column with numerical values using one hot encoding

work\_type\_transformer = make\_column\_transformer(

  (OneHotEncoder(), ['Game\_Type']),

  remainder='passthrough',

  verbose\_feature\_names\_out=False

)

# Load backup dataframe to maintain idempotency

dataframe = dataframe\_copy

work\_type\_transformed = work\_type\_transformer.fit\_transform(dataframe)

dataframe = pd.DataFrame(work\_type\_transformed, columns=work\_type\_transformer.get\_feature\_names\_out())

dataframe.head(20)

One-Hot Encoding is a technique used to convert nominal categorical variables, which have no intrinsic order, into a format suitable for machine learning models. This method involves creating binary columns for each category, representing the presence or absence of each category. However, to avoid multicollinearity a situation where independent variables are highly correlated with each other—it's common practice to drop the first column from the encoded set. This approach ensures that the encoded features remain linearly independent, thereby improving the stability and interpretability of the model.

By preparing the data in this manner, you ensure that the dataset is clean, structured, and ready for the modeling phase, with categorical variables properly encoded and irrelevant or problematic data removed.

## Insights Gained

During the data exploration and preparation phase, several valuable insights were derived, which can significantly influence the subsequent steps in the analysis:

1. Correlation Analysis:

The correlation matrix helps to understand relationships between features and the target variable, Average\_Rating, highlighting features with strong correlations as valuable for model inclusion. It also identifies potential multicollinearity issues, which can be addressed by removing redundant features or using techniques like Principal Component Analysis (PCA) to reduce feature redundancy and enhance model stability.

2. Histograms:

Histograms offer a visual overview of the distribution of numerical features, revealing data spread, central tendency, and skewness. They help detect skewness, indicating if transformations like logarithmic scaling are needed to normalize the data and enhance model performance. Additionally, histograms show the range of values for each feature, guiding decisions on data scaling or normalization.

3. Boxen Plots:

Boxen plots are effective for detecting outliers that could impact model performance, allowing for decisions on handling them, such as removal or transformation. They also provide insights into the distribution of values, especially in large datasets, guiding decisions on data transformation and scaling.

# Machine Learning Approach

Here's a more detailed explanation of each step in the machine learning approach:

1. Feature Scaling

Feature scaling ensures all independent variables contribute equally to the model by standardizing features to a mean of 0 and a standard deviation of 1 using StandardScaler. This process, crucial for algorithms reliant on distance metrics or gradient descent, prevents features with larger ranges from disproportionately influencing the model, leading to more balanced and accurate learning.

2. Data Splitting

Data splitting divides the dataset into training (80%) and testing (20%) sets to evaluate the model's performance on unseen data and prevent overfitting. The training set helps the model learn relationships between features and the target variable, while the testing set assesses how well the model generalizes to new data.

3. Model Selection

The Random Forest Regressor, an ensemble method that averages predictions from multiple decision trees, is chosen for its robustness and flexibility. Key hyperparameters like n\_estimators, max\_depth, and min\_samples\_split are fine-tuned to enhance the model's accuracy and ensure effective generalization.

4. Model Training

Model training involves fitting the Random Forest Regressor to the training data, allowing it to learn patterns by constructing multiple decision trees from different data subsets. The aggregated predictions from these trees improve accuracy and robustness, leveraging the collective insights from the ensemble.

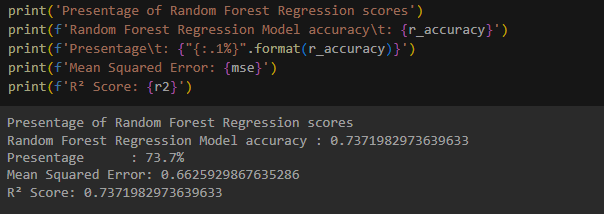
# Model and Performance Metrics

Prediction

The trained Random Forest Regressor is used to predict the Average\_Rating on the test set, allowing assessment of its performance on new, unseen data. By applying the predict method, the model generates predictions based on test features, which are then compared to actual values to evaluate the model's generalization and accuracy in real-world scenarios.

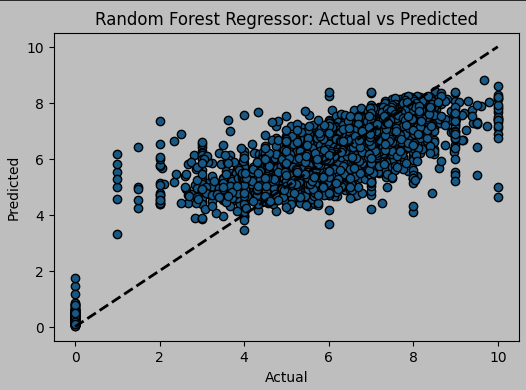
Evaluation Metrics

Key metrics like Mean Squared Error (MSE) and R² Score are used to assess the model's performance. MSE measures the average squared difference between actual and predicted values, indicating prediction accuracy. The R² Score reflects the proportion of variance explained by the model, with higher values showing better performance. Model accuracy is also considered, representing the percentage of correct predictions relative to the total.



Residual Analysis

Residual analysis examines the differences between actual and predicted values by plotting residuals against predicted values. Ideally, residuals should scatter randomly around zero, indicating unbiased errors. Patterns in the residuals may suggest areas where the model needs refinement or where alternative approaches could be explored.



# Discussion of Model Pros and Cons

The Random Forest Regressor offers significant advantages, including robustness to overfitting and the ability to handle both numerical and categorical data effectively. Its ensemble approach, which combines multiple decision trees, enhances generalization and reduces the risk of overfitting. Additionally, it provides valuable feature importance scores, helping with feature selection and understanding data relationships. The model is also scalable, capable of handling large datasets efficiently and can be parallelized to improve training speed.

However, the Random Forest Regressor has some drawbacks. Its complexity makes the model less interpretable, as understanding individual predictions is difficult due to the aggregated nature of decision trees. This complexity also leads to higher computational costs, with increased memory usage and longer training times when using large numbers of trees. Additionally, while feature scaling can be beneficial, it may obscure the interpretability of original features, making it harder to understand the impact of individual features on predictions.

# References

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-in-predictive-analytics-01c31897c1d4

# Appendices